14th International Conference on Information Fusion
Chicago, Illinois, USA, July 5-8, 2011

Supervised Learning in CINets

Paul Bruhn
The Applied Research Laboratory
The Pennsylvania State University
University Park, PA, U.S.A.
plb123@psu.edu

Dr. Jeffrey Weinschenk
The Applied Research Laboratory
The Pennsylvania State University
University Park, PA, U.S.A.
jjw15@arl.psu.edu

Abstract – ***Continuous Inference Networks (CINets), a form of multilayer fuzzy value networks, allow computation with fuzzy values in concise structures, are capable of universal function approximation, and are readily interpretable through natural language, aiding maintenance, modification, collaboration, and knowledge sharing. However CINets have been reliant on Subject Matter Expertise (SME) and manual tuning to realize optimal performance, limiting their applicability. With ONR support[i], ARL has developed a supervised learning process for CINets, capable of designing a CINet structure, and of optimizing an existing CINet structure. The CINet supervised learning process allows the automated development of data fusion, classification, and pattern recognition structures that are interpretable, modifiable, and concise. Performance of CINets developed with the supervised learning process is compared to that of Artificial Neural Network (ANNs), fuzzy logic rule set, and Bayesian network approaches.***

Keywords: Fuzzy Logic, CINet, Supervised Learning, Network Learning, Particle Swarm Optimization, Data Fusion, Classification.

1 Introduction

In the mid 1990s, a flexible data fusion, pattern recognition and classification tool termed Continuous Inference Networks (CINets) was developed at the Applied Research Laboratory at The Pennsylvania State University (ARL/PSU) to capture Subject Matter Expert (SME) domain knowledge for torpedo and anti-torpedo guidance and control automation.[1] CINets have been used for problems including shipboard damage control, medical anesthesia monitoring, WMD agent detection, and condition-based maintenance. CINet structures are multi-level fuzzy logic networks, and have some advantages of both fuzzy logic systems and Artificial Neural Networks (ANNs). Like fuzzy logic systems, the CINet technique allows the use of human-intuitive approximate descriptions of a problem domain to build an effective structure.[2] Like multi-layer perceptrons, the formalism can be used to build compact and fast structures, scalable to large Multi-Input Multi-Output (MIMO) problems, while avoiding the rule explosion problems that affect fuzzy rule systems[3]

CINets also maintain features common to both fuzzy systems and ANNs. The technique can be be shown to possess the property of universal functional approximation. CINet structures are implemented using a mixture of scalar and vector mathematics, and so tend to be efficient enough to use in real-time systems. In the worst case (a CINet with complete interconnection of all nodes) execution time is factorial with the number of nodes, but typical SME-designed CINets are concise networks with reduced interconnections, and therefore have complexity that approaches a linear function of the number of inputs. Figure 1 below shows the structure of a section of a CINet classifier for a problem that will be discussed in sections 2 and 3 of the paper.

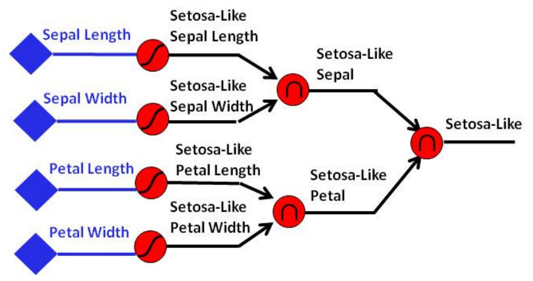

Figure 1: Iris Classifier CINet Setosa Subnetwork

In the past, CINet structures have been designed and tuned by SMEs using a simulation-based design process; a labor-intensive process for

[i] Work sponsored under ONR Contract # N00014-05-G-0106/0113. Information in this paper does not necessarily reflect the position or policy of the Office of Naval Research or the government of the U.S.A.; no official endorsement should be inferred.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE **JUL 2011**	2. REPORT TYPE	3. DATES COVERED **00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE **Supervised Learning in CINets**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Pennsylvania State University,Applied Research Laboratory,University Park,PA**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 14th International Conference on Information Fusion held in Chicago, IL on 5-8 July 2011. Sponsored in part by Office of Naval Research and U.S. Army Research Laboratory.

14. ABSTRACT
Continuous Inference Networks (CINets), a form of multilayer fuzzy value networks allow computation with fuzzy values in concise structures, are capable of universal function approximation, and are readily interpretable through natural language, aiding maintenance modification, collaboration, and knowledge sharing. However CINets have been reliant on Subject Matter Expertise (SME) and manual tuning to realize optimal performance, limiting their applicability. With ONR support[i], ARL has developed a supervised learning process for CINets capable of designing a CINet structure, and of optimizing an existing CINet structure. The CINet supervised learning process allows the automated development of data fusion, classification, and pattern recognition structures that are interpretable modifiable, and concise. Performance of CINets developed with the supervised learning process is compared to that of Artificial Neural Network (ANNs), fuzzy logic rule set, and Bayesian network approaches.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**8**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

most problems of interest. Ad-hoc CINet creation also requires that an SME have an adequate level of knowledge to structure a CINet that will perform well for all variations of the problem domain. By utilizing machine learning techniques, CINets can be constructed via a supervised learning approach. With support from the Office of Naval Research (ONR) Code 333, ARL/PSU has developed a supervised learning process to produce CINet objects using labeled training data. Introduction of supervised learning for CINets extends their use to a much broader set of applications: automated network construction and tuning with labeled data sets, on-line network adaptation, and knowledge discovery, all while retaining the intrinsic benefits of the CINet approach. Section 2 of this paper will describe the approach used to provide supervised learning for CINets. Section 3 will compare performance of CINets developed using the supervised learning process to CINets developed manually and to other classification techniques, and section 4 will present conclusions on the approach.

2 CINet Supervised Learning Approach

Using supervised learning to both choose the fuzzy logic operators for the connective nodes and to set the input weights can greatly reduce the level of effort required to develop CINets. The supervised learning process has three stages:

1. Set the dimensions (number of layers and number of nodes in each layer) of a fully-connected network

2. Optimize the weights and connective fuzzy operators of the network

3. Remove nodes and edges that do not affect the CINet ouput ("Pruning").

The third stage of the process delivers a concise CINet structure that will require fewer computational resources in operation, but is also easier for researchers to study and gain insights into the problem domain. In cases where an initial network structure is available (i.e., an SME has designed a CINet for the problem), the supervised learning process can be used to optimize the weights and fuzzy operators within this structure. In this case, only the second stage of the supervised learning process is required. However, the third pruning stage of the process may be able to identify segments of an SME-designed CINet that are unnecessary. In this way, the supervised learning process may be able to add to even the SME's knowledge of the problem. Section 2.1 will discuss the significance of weight values in a CINet, and how the supervised learning process uses the weights for structural learning and optimization. Section 2.2 will present the additions to the CINet methodology necessary to support supervised learning. Section 2.3 will discuss the application of the Particle Swarm Optimization (PSO) algorithm for CINet supervised learning, and section 2.4 will describe the CINet structural reduction stage of the supervised learning process.

2.1Weight Values in CINets

CINet structures are directed weighted graphs, where the nodes are either fuzzy membership functions or fuzzy logic operators, and the edges are fuzzy membership values. A CINet is typically structured as: a set of input sources, a set of membership functions as the first layer of the network, and one or more intermediate layers consisting of a set of connective nodes, producing one or more output fuzzy values. As an example, Figure 1 above shows a section of a CINet for the Fisher iris classification problem. Fuzzy logic operators can include fuzzy union and intersection operations ("OR" and "AND"), but negation and M out of N ("MOON") operations have been employed. The CINet technique does not constrain the specific choice of operator implementation; standard max-min and product-sum implementations have been used, as well as others.[4]

Weights on the input edges are real numbers in [0 1]. The weights on the input edges to a connective node represent the relative significance of the input fuzzy values to the output fuzzy value from the node. The specific impact of varying an input weight will depend on the type of fuzzy operator and on the fuzzy logic implementation chosen. Generally, a weighted fuzzy union operation will be the fuzzy union of the intersections of each input value with its corresponding weight, as diagrammed in Figure 2. The figure also shows the diagram of a weighted fuzzy intersection operation, which is the fuzzy intersection of the unions of its inputs with the complements of its corresponding weights[5].

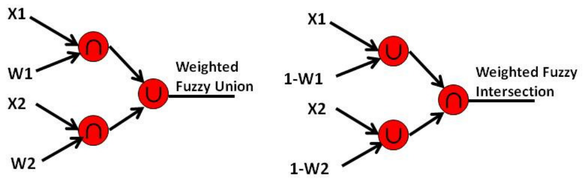

Figure 2: Weighted Fuzzy Union and Intersection

Some examples will be used to show how the impacts of varying the weights on the inputs are dependent on the choice of fuzzy logic implementations used for the base intersection and union operators. When max/min fuzzy logic is chosen, then the weighted fuzzy intersection operation becomes:

$$min[max(x_1, 1-w_1), max(x_2, 1-w_2)] \quad (1)$$

and the weighted union becomes:

$$max[min(x_1, w_1), min(x_2, w_2)]. \quad (2)$$

For the weighted intersection operation, this means that the weight on a given input places a threshold on how low that input can drive the output; i.e., if an input has a corresponding weight of 0.3, and all other inputs are 1.0, then the output will be equal to the first input *unless* it is less than 0.3, in which case the output will equal 0.3. Conversely, for the weighted union operation, a weight will limit how high the corresponding input can drive the output of the operator.

If sum/product fuzzy logic implementations are used instead of max/min logic, then the thresholding behavior of the weights will still be present, but in addition there will be a scaling of the sensitivity of the output to the corresponding input, proportional to the weight. For a sum-implemented union operation, if all inputs but the first are 0.0, then the output will be the product of the first input by its corresponding weight. So in that case, if the weight on the first input is 0.7, then the input will be unable to drive the output higher than 0.7, but the output will be proportional to the first input across its entire range. For all choices of fuzzy logic implementation, when the weight on an input goes to 0.0, then the output should be completely independent of that input, so that the input edge can then be removed from a CINet graph without affecting the input-output behavior of that CINet. This is the main mechanism used for graph structural learning in the CINet supervised learning process.

2.2 Weighted Fuzzy MOON Operators

While the supervised learning process is able to specify the structure of a CINet using input weights, as part of the optimization, the process also needs to be able to determine the operator type for the nodes. Fuzzy AND and OR operations are the two most-frequently used operators in manually constructed CINets, but the operation of one is strongly dissimilar from another. Including some form of weighted fuzzy operator with intermediate operation between a fuzzy AND and an OR would lead to a more finely graduated solution space for the optimization algorithm. This in turn should improve the search performance of non-exhaustive search algorithms.

Fuzzy logic systems researchers have previously identified connective operators that offer intermediate behavior between a fuzzy OR and a fuzzy AND operation.[6] Logical M out of N ("MofN") operators are intermediate operations to crisp logic AND and OR operators. They return "True" when at least M out of their N inputs are "True." When M = 1, they are equal to logical OR operators, and when M = N, they are equal to logical ANDs. Logical MofN operators can be composed of subnetworks solely of logical AND and OR operators. Fuzzy MOON operations are analogous to crisp logic MofN , in that when at least M out of their N inputs are "high," then their output will also be "high." Similar to logical MofN, fuzzy MOON operations can be composed of subnetworks of fuzzy OR and fuzzy AND operations. The CINet supervised learning process requires a particular formulation of a weighted fuzzy MOON operator as the connective function for all connective nodes in a CINet structure:

$$MOON(\mathbf{inputs},\mathbf{weights},M) = fuzzy\ OR(\boldsymbol{\Omega_M},\boldsymbol{\Psi_M}),$$

where

$$\boldsymbol{\Omega_M} = fuzzy\ AND[\mathbf{combination(inputs},M), \mathbf{combination(weights},M)], \quad (4)$$

and

$$\boldsymbol{\Psi_M} = fuzzy\ AND[\mathbf{combination(weights},M)]. \quad (5)$$

In the prior equations, a **boldface** identifier represents a vector quantity (either a vector of values or a vector of functions). The ***combination(X,M)*** operation returns a vector of unique sets of *M* members of ***X***; e.g., with ***X***=[A,B,C] and *M*=2, then ***combination(X,M)*** *= {[A,B],[B,C],[C,A]}*. So, the weightedMOON is a fuzzyOR of the weighted fuzzyANDs of the unique combinations of M of the inputs, with the fuzzyAND results weighted by the fuzzyANDs of the corresponding unique sets of weights on the inputs. For use in the CINet supervised learning process, the weighted fuzzy MOON operator must simultaneously satisfy three requirements:

a) weighted fuzzy MOON must equal a weighted fuzzy OR when M =1

b) weighted fuzzy MOON must equal a weighted fuzzy AND when M = N

c) weighted fuzzy MOON must be independent of an input with a corresponding weight of 0.0; e.g.,

$$MOON([x, y, z], [w_x,w_y,0.0], 2) = MOON([x,y],[w_x,w_y], 2). \quad (6)$$

For requirements a), b), and c) to be simultaneously satisfied with the formulation of the weighted MOON given in (3), it is necessary to explicitly define that a weighted fuzzy AND or OR operator, with only a single fuzzy membership value input, will give the input value as the output, regardless of the weight on that single input. This constraint is logically consistent with a model where the weight on an input to a fuzzy operator represents the *relative* significance of that input; if an operator only has a single input, that input is by default the most significant, and is the only factor determining the output value of the operation. It must be noted that, outside of individual input weights being bounded between [0 1], there are no further constraints on the individual or collective input weights of the weighted MOON operator, and they can be freely set in accord with the SME's evaluations of relative significance, or to improve performance of the CINet. This is a difference from the constraints on setting weight values in operators such as the Weighted Modified Hamacher Operator (WMHO), where input weights are strictly set so that multi-level fuzzy networks behave identically to equivalent "flat" n-input single t-norm operations[7]. In addition to ensuring equivalent behavior, WHMO weights give greater significance to lower-level inferences with more inputs, but are not freely selectable by the network designer.

2.3 Design and Optimization of CINets using Particle Swarm Optimization

Using the weighted MOON operator, the supervised learning process must set the value of M of each connective node, and the weight for each input edge to each connective node. The values for the input weights are continuous variables in [0 1], but the M values must be positive integers. Also, for a particular connective node with N inputs, values of M > N are not applicable. To avoid implementing a mixed-integer or hybrid continuous-discrete search algorithm, a function is developed that maps a continuous value $\widehat{M}$, defined on the interval [0 1], into the integer M. The mapping to determine M from $\widehat{M}$ and N is

$$M = 1+round[\widehat{M}*(N-1)] \tag{7}$$

where *round*(x) is rounding to the nearest integer. The reverse mapping is

$$\widehat{M} = \begin{cases} \frac{\mathrm{M}-1}{\mathrm{N}-1}, & N > 1 \\ 1.0, & N = 1 \end{cases} \tag{8}$$

so a $\widehat{M}$of 0.0 corresponds to a fuzzy union operation, and 1.0 corresponds to a fuzzy intersection operation. By replacing the M values with continuous values, the solution space becomes a hypercube with length 0.0 → 1.0 in all dimensions.

The Particle Swarm Optimization (PSO) algorithm, first published in 1995, is a guided stochastic search algorithm, popular for its simplicity and general suitability for a wide range of optimization problems.[8] For supervised learning for CINets, each particle is a complete vector of $\widehat{M}$ and input weight parameters, specifying a potential solution CINet. As the PSO algorithm proceeds, the parameter vector particles should converge towards a global optimum; i.e. a set of $\widehat{M}$ and input weight settings for an optimally performing CINet for the problem in question.

The user must define the performance function that the PSO algorithm uses to evaluate the performance of the particles in each cycle of the optimization. This means that the CINet supervised learning process becomes applicable to many types of pattern recognition or data fusion problems when an appropriate performance function is selected. An example candidate performance function for a regression problem is:

$$e^{-(||\boldsymbol{Labels(X)\text{-}Outputs(X)}||)}; \tag{9}$$

i.e., performance is the *e* to the negative difference between the label outputs for examples **X** and the CINet outputs for examples **X**. The performance approaches 1.0 when the difference between the label values and the CINet output is small for all examples. For a classification problem, a possible performance function is:

$$\#correct\_classifications\,/\,\#examples, \tag{10}$$

where a classification is correct if the output of the CINet for the correct class is the highest of all the CINet outputs for the example. Performance functions can also be made directly dependent on the optimization parameters themselves. For example, for a problem where it is important that the final network design have as few remaining components as possible, a performance function that penalizes input weight parameters not set to 0.0 or 1.0 could be used. The supervised learning process would then tend to find optimal solution CINets with only full-weight input edges or zero-weight edges. The many zero-weight edges could then be discarded in the reduction stage.

The CINet supervised learning process allows the user to modify the standard PSO parameters to control PSO discovery and refinement during optimization. The user is also able to determine the number of particles to use during an optimization trial, and the exit conditions that terminate a given trial. As with any stochastic search optimization, the number of iterations necessary to reach an acceptable optimum cannot be *a priori* determined, so PSO trials terminate after some pre-set cycle count. The user can also set a required minimum necessary performance level a minimal level of improvement *d* necessary to continue more than *P* cycles (if after *P* cycles, the performance of the global best has not improved by *d* or greater, then the trial will terminate), or a minimum allowable radius of the particles (the trial will end when all of the particles are within a radius *r* of each other). All of the potential exit criteria are available to use simultaneously.

One feature of the PSO is that the particles are not bound to remain within an initial region of the search space. For many optimization problems, this can be considered an advantage (the user does not need to be able to initially determine the region of the search space in which the global optimum lies), but for CINet supervised learning, the only valid solutions are within the hypercube between 0.0 and 1.0 in all directions. On each cycle, any particle that has moved outside of the hypercube is adjusted to lie on the point of the surface of the hypercube where it exited. In an early design of the supervised learning process, the hypercube surfaces were made "elastic;" i.e., particles that were returned from outside the hypercube had their velocities reversed. However, this caused the motions of the particles within the search space to be unstable – particles would soon oscillate from surface to surface, with no discovery of optima in the interior space. The final version of the process sets the velocity of errant particles to zero, which results in better subsequent search of the interior of the solution space.

2.4 CINet Structure Reduction

Pruning of CINet network components that make no contribution to the output will result in a CINet structure closer to the minimal network necessary to provide correct results. This will improve the generalization of the resulting CINet to new input examples, although it will not affect overtraining during the optimization stage. Because the MOON operator calculates the fuzzy AND operator for all unique combinations of M inputs, the number of suboperators calculated in the weighted MOON operator will equal 2*[(N-1)!] in the worst case (when M = ½ N or M = ½(M+1)). This means that CINets with connective nodes that each have many input edges will also tend to be computationally complex. Since the supervised learning process is initialized to full interconnection between layers, this in turn means that the time to complete the optimization phase of the process will tend to increase factorially with the number of nodes in each layer. The pruning phase, by eliminating unnecessary edges, can dramatically improve the execution time for the final CINet structure; an advantage for CINets intended for application in real-time systems. Finally, this reduced structure can also be more readily studied to gain insights into the problem domain.

The reduction phase of the CINet supervised learning process removes three types of components:

1- Input edges with corresponding weights set to zero.

2- Connective vertices with a single fuzzy input.

3- Vertices, other than output vertices for the CINet structure, with no output edges (which also removes all of its input edges).

In the first case, if the connective node that takes the edge as an input has at least one other input, then the zero-weight edge is removed from the network, and the connective node that had the zero-weight edge as an input has its N value reduced by one. The strictness value for the node is held constant, so the M value for the node may or may not be reduced. In the second case, any connective node with only a single input edge is removed from the network. All of the edges that had received the output of the removed node are instead connected to the node that was the source of the removed node's input edge.

In the third case, any node that is not an output node for the CINet structure and with no output edges is removed from the network. It is possible for dead-end nodes to be created by the removal of other nodes but remain in the network, using the current single-pass version of the CINet reduction algorithm. Multiple passes, or an explicitly recursive reduction algorithm, would be necessary to guarantee the removal of all unnecessary network components. In the case when supervised learning for CINets is used to tune a structure initially provided by a SME, that structure may already clearly and concisely capture the inputs, outputs, and intermediate properties necessary to provide correct results for events in the problem domain. When it is not, the supervised learning process will be able to prune nodes and edges that prove extraneous when the CINet is optimized.

3 Performance Comparison

To evaluate the performance of the resulting CINet structures, the supervised learning process was applied to develop CINets for a common reference problem. The Fisher iris classification problem is one of the seminal studies in multicriteria classification, and continues to be used as a standard reference problem in classification.[9] The data set consists of measurements for 50 samples each from three iris species; iris setosa, iris versicolor, and iris virginica. The measurements for each sample are petal length and width, and sepal length and width. The species' sample distributions overlap, and there are a few outlier samples that fall substantially within the distributions of other species.

So as to help bound the dimensions specified for the supervised learning process, a CINet was constructed and tuned by an SME; i.e. the authors, after studying the Fisher iris data set. Weighted sum/product operations were used for the fuzzyOR and fuzzyAND operations:

$$fuzzyAND(\boldsymbol{x},\boldsymbol{w}) = \prod_{i=1}^{n}(1 - w_i + w_i * x_i), \quad (11)$$

and the fuzzyOR formulation:

$$fuzzyOR(\boldsymbol{x},\boldsymbol{w}) = 1 - \prod_{i=1}^{n}(1 - w_i * x_i). \quad (12)$$

Figure 1 in Section 1 shows the structure of the iris setosa-species-specific subnetwork of the SME CINet. There were identically-structured subnetworks for the iris versicolor and iris virginica species. The classification given to an example by the CINet would be the iris species with the highest output from its corresponding subnetwork. After four cycles of manual parameter tuning, performance was 96.0% correct classifications over all samples.

For training of CINets for the iris classification problem, a performance function that emphasized classification was used: a CINet is considered to have correctly classified a sample if the network output for that species is greater than the output for the CINets for either of the other two species, even if the other outputs are only slightly less than the output for the correct species. The performance score over a set of samples is simply

$$score = \frac{num.\,correct\ classifications}{total\ number\ of\ samples} \quad (13)$$

The data set was randomly partitioned into evenly sized training and testing sets, with the proportions of samples from each species preserved in each set (i.e., both the training and the testing sets had 25 iris setosa samples, 25 versicolor samples, and 25 virginica samples). Since the iris data set contains one or two examples for each species that fall far outside the normal distribution for that species, a 50%/50% partition of the data was chosen to make it impossible for either subset to be made up entirely of outlier examples. For each of three different CINet initializations, 10 independent supervised learning trials were run, and statistics gathered on training performance over these trials.

The first evaluation used the domain-expert-specified baseline iris classifier structure. The second evaluation used a general network structure with the same number and arrangement of vertices, but with full connection between one layer and the next (i.e. each of the 12 input membership function vertices was connected to all 6 intermediate vertices, and each of the intermediate vertices was connected to all output vertices. The final evaluation used a structure with each input vertex connected to only two intermediate vertices, as a middle ground between the structures used in the 1st and 2nd evaluations. This third structure is representative of the kinds of assumptions that a researcher might be able to make about the associations of input data in a problem domain, even if they don't have a sufficient level of expert knowledge to completely structure a CINet for the problem. Figure 2 shows the initial network structure for the final evaluation. All of the structures were evaluated with 20 particles per trial. The second and third structures were also evaluated with 50 particles per trial.

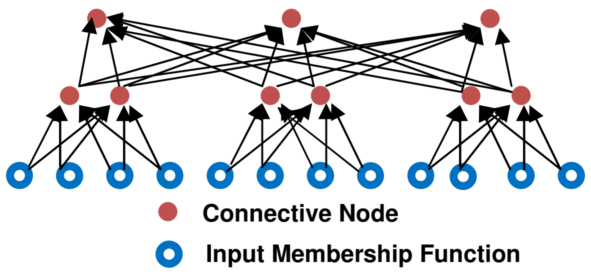

Figure 2: CINet Structure for Third Evaluation

Table 1 shows the training performance of the process in the 3 evaluations. The "Max" and "Min" rows in table 1 are the maximum and minimum values over all 10 trials, with performance values being the best performance on the final cycles of the independent trials. Trial time statistics are on a modern dual-core 2.4 GHz desktop processor, with a single-threaded supervised learning process and 4GB of RAM available. Table 2 compares the best achieved classifier performances against other published methods, including ANNs, Bayesian networks or fuzzy rule set classifiers.[10],[11],[12] Like this study, the ANN study uses a 50%/50% partition of training/testing data, but the Bayesian network study uses a 90%/10% partition, and the fuzzy rule set study uses a Leave-One-Out (L1O) (149/1) partition.

Evaluation		1st	2nd		3rd	
Parameters		27	99		51	
Particle Count		20	20	50	20	50
Score (%)	Max	97.3	72.0	97.3	96.0	98.7
	Min	93.3	54.7	86.7	33.3	93.3
Iteration	Max	46	18	100	19	98
	Min	22	11	19	11	16
Trial Time (s)	Max	267	$6.01e^3$	$1.08e^5$	269	$3.22e^3$
	Min	87	$1.08e^3$	$6.32e^3$	136	653
Nodes Reduced	Max	0	0	0	0	0
	Min	0	0	0	0	0
Edges Reduced	Max	7	18	35	19	17
	Min	5	8	34	13	16

Table 1: Iris Problem Training Statistics

ANN	Bayesian Network	Fuzzy Rule Set	CINET Sup. Learning		
			1st	2nd_50p.	3rd_50p.
97.3	95.5	96.7	97.3	97.3	98.7

Table 2: Maximum Iris Classification Scores for CINETs and Other Methods

For the first evaluation, the supervisory learning process achieved performance close to or surpassing the "manually" constructed CINets on all trials. In most cases, the process identified an optimal solution in as few as two iterations, but one trial took 46 iterations. That trial had the highest classification performance against the training set (98.7%), but relatively low performance against the test data (93.3%); i.e. the process had overtrained against the training set. The process does have a parameter to stop training when a performance threshold is reached, but that threshold was set to 100% for all trials.

The second evaluation experienced far longer training times, partially due to the higher number of training parameters, but principally due to the number of nodes with high numbers of inputs in the structure. With 20 particles used in training, the second evaluation produced poor classification scores from the final CINets. The process was prone to settling in local optima, such as networks that correctly classified two of the species, but never detected the third species. Increasing the number of particles to 50 enabled the process to adequately cover the solution space and identify optima on par with those seen in the first evaluation, with the tradeoff of longer training times.

The third evaluation had good classification performance with 20 particles on some trials, but the range of performances between trials was significantly larger than for the first evaluation. The time required to train the CINets was on the order of that required for the first evaluation. With 50 particles used, the maximum and mean performances were higher than those of the first evaluation. Figure 3 shows an example of the final optimum CINet structure after reduction for one of the trials in the third evaluation.

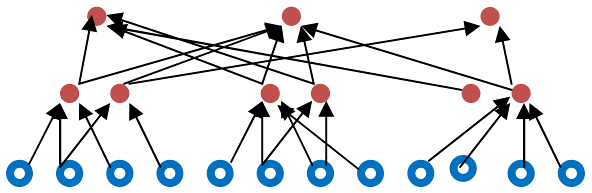

Figure 3: Reduced CINet Structure for Third Evaluation

Note the "unsupported" fifth intermediate node in Figure 3; a weighted MOON operator with zero weights on all inputs will always produce an output of zero. In these cases, the reduction stage replaces the connective node with a constant 0.0 source node. The effect of this constant source node and the weight on the edge going from it to the second output node is similar to the bias term on a node in a multilayer perceptron. Compared to the SME-provided initial structure used in the first evaluation, this final structure contains many edges linking nodes driven by features for one species to nodes for different species. Since the classification performance of this optimized network is higher than that achieved with the SME-designed structure, a user could study the more-connected structure and gain insights into the classification problem.

4 Conclusions

ARL/PSU has developed a supervised learning process to construct and optimize CINet structures for classification, regression, and data fusion problems from labeled training data. Through the use of fuzzy operators that can be read off as natural language by an observer, human interpretability is preserved for CINets constructed by the process. Since the process also includes a reduction stage that acts to refine trained CINets into more concise, minimal structures, resulting CINets are more general with respect to novel inputs, have reduced computational complexity, and are even more readily interpretable. The supervised learning process for CINets can be used to determine a CINet structure through training a large, fully connected network, and then reducing that network by removing components that have no impact on the CINet output. The process is also demonstrably useful for setting the parameters of an SME-designed CINet structure to improve performance.

Without altering the CINet supervised learning process, modifying the PSO training parameters may cause better exploration of the solution space. The supervised learning process itself could be improved by a number of means:

1- Training time could be improved by identifying a different connective node operator that would meet the requirements in section 2.1, but avoid factorial scaling of complexity with the number of inputs.

2- Utilize performance functions that explicitly penalize some of the low-performing suboptimal solutions, such as making one class "unclassifiable." Additionally, concepts from fuzzy logic analysis, such as vagueness, could be incorporated into classification performance functions to encourage greater separation in the CINet outputs, thus making "no-class" outputs less likely.[13]

3- Although "canonical" PSO has been seen to have good general search performance, cooperative learning PSO training might improve training performance in solution spaces with a high number of dimensions, by training subsets of the parameters independently and then combining the subsets at each iteration to evaluate overall performance.[14] Backpropagation, or other fast optimization techniques utilizing the error gradient, could replace PSO in problems where the performance surface has a small number of optima and no discontinuities.

4- Recursive pruning could guarantee that all removable components are removed from the final network of a CINet, guaranteeing maximum generalizability and interpretability, and minimal complexity, for a given optimal parameterization.

5- By using a generative network learning approach, instead of the reductive approach presented here, CINet training times might be significantly reduced. Generative approaches start with a small number of nodes and then add edges and nodes until a sufficient structure is constructed. As a result, over-connected nodes (that scale factorially in complexity) could be avoided, and the total number of nodes would tend to be smaller during training, linearly reducing complexity and training time.

References

[1] J. A. Stover, D. L. Hall, R. E. Gibson, "A Fuzzy Logic Architecture for Autonomous Multisensor Data Fusion," *IEEE Trans. on Industrial Electronics*, Vol. 43, No. 3, June 1996.

[2] J. J. Weinschenk, W. E. Combs, R. J. Marks II, "Avoidance of rule explosion by mapping fuzzy systems to a disjunctive rule configuration," *IEEE Int'l Conference on Fuzzy Systems*, St. Louis, MO, pp 43-48, 2003.

[3] J. J. Weinschenk, "Complexity Reduction in Fuzzy Inference Systems," Ph.D. Dissertation, University of Washington, 2005.

[4] G. J. Klir, B. Yuan, Fuzzy Sets and Fuzzy Logic, Prentice Hall, NJ, 1995.

[5] C. A. Janes, Weighted Logic Decision Modeling Theory and Application, M.S. Thesis, The Pennsylvania State University, 1994.

[6] R. R. Yager, "Families of OWA Operators," *Fuzzy Sets and Systems*, Vol. 59, pp. 125-148, 1993.

[7] U. Scheunert, P. Linder, H. Cramer, "Multi Level Fusion with Fuzzy Operators using Confidence," *9th Intl. Conf. on Information Fusion,* Florence, Italy, July 2006

[8] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," *Proceedings of IEEE International Conference on Neural Networks*, Piscataway, NJ, pp. 1942-1948, 1995.

[9] R.A. Fisher, "The Use of Multiple Measurements in Taxonomic Problems," *Annals of Eugenics,* Vol 7, pp. 179–188, 1936.

[10] Di Wang, "Fast Constructive-Covering Algorithm for Neural Networks and its Implement in Classification," *Applied Soft Computing*, Vol. 8, pp. 166-173, 2008.

[11] S. B. Kotsiantis, P. E. Pintelas, "Logitboost of Simple Bayesian Classifier," *Informatica*, Vol. 29, pp. 53-59, 2005.

[12] H. Ishibuchi, T. Nakashima, T. Morisawa, "Voting in Fuzzy Rule-Based Systems for Pattern Classification Problems," *Fuzzy Sets and Systems*, Vol. 103, pp. 223-238, 1999.

[13] Y. Yuan, H. Zhuang, "A Genetic Algorithm for Generating Fuzzy Classification Rules," *Fuzzy Sets and Systems*, Vol. 84, pp. 1-19, 1996.

[14] Van Den Bergh, A. P. Engelbrecht, "Cooperative Learning in Neural Networks Using Particle Swarm Optimizers," *South African Computer Journal*, Vol. 26, pp. 84-90, 2000.